Information Hiding Through Errors: A Confusing Approach *

Mercan Topkara Umut Topkara Mikhail J. Atallah
Department of Computer Science
Purdue University
West Lafayette, IN, 47906, USA
{mkarahan,utopkara,mja}@cs.purdue.edu

ABSTRACT

A substantial portion of the text available online is of a kind that tends to contain many typos and ungrammatical abbreviations, e.g., emails, blogs, forums. It is therefore not surprising that, in such texts, one can carry out information-hiding by the judicious injection of typos (broadly construed to include abbreviations and acronyms). What is surprising is that, as this paper demonstrates, this form of embedding can be made quite resilient. The resilience is achieved through the use of *computationally asymmetric transformations* (CAT for short): Transformations that can be carried out inexpensively, yet reversing them requires much more extensive semantic analyses (easy for humans to carry out, but hard to automate). An example of CAT is transformations that consist of introducing typos that are *ambiguous* in that they have many possible corrections, making them harder to automatically restore to their original form: When considering alternative typos, we prefer ones that are also close to other vocabulary words. Such encodings do not materially degrade the text's meaning because, compared to machines, humans are very good at disambiguation. We use typo confusion matrices and word level ambiguity to carry out this kind of encoding. Unlike robust synonym substitution that also cleverly used ambiguity, the task here is harder because typos are very conspicuous and an obvious target for the adversary (synonyms are stealthy, typos are not). Our resilience does not depend on preventing the adversary from correcting without damage: It only depends on a multiplicity of alternative corrections. In fact, even an adversary who has boldly "corrected" all the typos by randomly choosing from the ambiguous alternatives has, on average, destroyed around w/4 of our w-bit mark (and incurred a high cost in terms of the damage done to the meaning of the text).

1. INTRODUCTION

Natural language watermarking traditionally targets grammatical, or even edited text where preserving the grammaticality and style is one of the main concerns.[1, 2] The concern for quality of the watermarked text forces systems to perform at a low embedding bandwidth and to put emphasis on the accuracy of natural language processing components.

However, a large percentage of daily exchanged digital text is in the form of e-mails, blogs, text messages, or forums; which we will call *cursory text*. This type of text is usually written spontaneously and is not expected to be grammatically perfect, nor to comply with a strict style. The freedom from being error-proof and from following a style, creates an opportunity for improved information hiding by applying completely new approaches tailored for cursory text, or by adapting the existing mechanisms that were proposed for edited text.

It is possible to use many idiosyncrasies of cursory text for information hiding, by modifying them or judiciously injecting them to the text. Such idiosyncrasies include:

- Unintentional typographical errors (character typing errors such as "teh" instead of "the").
- Well-known abbreviations and acronyms (e.g. using "ur" instead of "you are" or "omg" for "oh my god", "b4" for "before").
- Transliterations such as leet-speak (e.g. "l33t" for "leet"), pig latin, gyaru-moji or inversion of syllables such as verlan (e.g. "my nopia is kenbro" instead of "my piano is broken").

*Portions of this work were supported by Grants IIS-0325345 and CNS-0627488 from the National Science Foundation, and by sponsors of the Center for Education and Research in Information Assurance and Security.

- Free formatting, such as unnecessary carriage returns or arbitrary separation of text into paragraphs, or varying line sizes.
- Usage of emoticons to annotate text with emotions or attitudes (e.g. ":)" for annotating a pun).
- Colloquial words or phrases (e.g. "gonna" or "ain't nothin"').
- Jargon specific to the age or interest group (e.g. "DCT" is used for "Discrete Cosine Transform" in engineering jargon, and it is used for "Divine Command Theory" in philosophy jargon.)
- Free usage of capitalization and fonts for richer annotation (e.g. "I AM joking").
- Mixed language use, where words from several different languages are used together in the same text (e.g. "We always commit the same mistakes again, and 'je ne regrette rien'!").
- Replacing native characters of an alphabet with latin characters such as writing in "faux cryllic" or writing "sh" instead of letter "ş" in Turkish.
- Grammatical errors.

In this work, we focus on using the typographical errors (henceforth referred to as *typos*), broadly construed to include the above-mentioned acronyms and abbreviations, for increasing the bandwidth of natural language information hiding. When resilience is important (as in watermarking), we make use of ambiguity to make it harder for the adversary to correct the typo. To illustrate this, consider the following example, based on the "lol" acronym that is uncommon, if not unacceptable, in formal text, but very common in cursory text. According to Wikipedia, "lol" has 17 different meanings that depend on the context in which it is used, including the following few:

- Internet slang: "lots of laughs".
- Legal and financial texts: "limits of liability".
- Culinary texts: "Land O Lakes" butter.
- Travel texts: "Derby Field Airport" (less strange than "ord" for O'Hare airport – Derby Field is located in Lovelock, Nevada)

Our current implementation has not yet automated the use of hard-to-reverse acronym substitutions such as "lol" (we used it as an example because it has such a high degree of ambiguity[†] to a text analysis software, yet practically no ambiguity for a human reader).

As mentioned above, another way to introduce ambiguity is using latin characters while writing in a non-latin language. The short sentence "iyi isitir" written in ASCII characters, has three different meaningful and grammatical mappings in Turkish script. One mapping is "iyi işitir" ("he/she/it hears well"), another is "iyi ısıtır" ("it heats well"), and the last one is "iyi ışıtır" ("it illuminates well"). This ambiguity occurs due to the many-to-one mapping of Turkish characters to ASCII characters. In its context, a human reader would not have any problem in disambiguating the correct Turkish script of above phrase when written in ASCII. Even though currently available automatic "de-asciifying" techniques[3] have an impressive average case performance on real text, they will not resolve the ambiguities that have been injected with the intention to confuse an automated "de-asciifying" system, such as the one in the above example.

Typos have the advantageous property of being common to all types of cursory text, e.g. emails, text messages, forum posts, etc.; hence an information hiding system based on this notion has a wide range of applications. They are usually injected into the text by their authors as a result of speed typing (e.g "teh" instead of "the"), or incorrect spelling knowledge (e.g. "principal" instead of "principle" or "tommorow" instead of "tomorrow")

[†]The list of different meanings of "lol" can be made even longer – "acronymfinder.com" lists 62 different meanings for "lol".

among other reasons. Typos can occur at any part of the text. However some words are naturally harder to spell or type with keyboard, hence typos occur more frequently with them.

Spelling checkers usually use the edit distance of typos between vocabulary words to suggest corrections. In some cases several corrections are viable for the typo, and spelling correctors use additional side information such as previously observed regularity of typos and models of the underlying natural language or similarity of pronunciation to prioritize this correction list.[4, 5] The regularities in typos are usually due to proximity of key locations on the keyboard, or mental proximity of syllables or words resulting from phonetic similarity. See Section 6 for more information about spell checkers.

Humans are usually better than automated spelling correctors in typo correction, for this reason, spelling correctors are usually insufficient to completely correct typos, and the author's intervention is required for the final decision. Human fluency in spelling correction also plays a role in the implicit correction of typos by readers of cursory text. Humans use a combination of pattern matching (e.g., resolving "imlipcit" stands for "implicit") and a wide variety of side information that surpass the boundaries of one message exchange, such as shared experience with the author (e.g., a reference to "Jerry" as "Gary"), and real life knowledge ("bear footprint" caption under a human bare footprint picture).

While designing an information hiding system for cursory text using the typographical errors, the following challenges should be taken into account:

- Preserving the value of the document. Even though the editorial quality, grammaticality or fluency of text are less of a concern in cursory text, the result of the information hiding process still has to be easily comprehensible by a human reader. Furthermore, some portions of the text may not tolerate *any* variations since they are central to the meaning of the cover text, such as the meeting place in a memo.
- Automatic spelling checkers can be used to reduce the possibility of using typos for information hiding.
- The adversary might have access to a good model of cover text including a model of the text that was previously generated by each author. The message exchanges can be strictly monitored, such as in company mail accounts, or in blogs.
- The embedding bandwidth in natural language text is lower when compared to that of image, video or audio, i.e., the number of words or sentences are comparable with the message length. For this reason the hidden messages are embedded more densely in the cover text documents when compared to cover documents of other media types.

We have designed two methods of embedding, one for an active adversary (which can be used for watermarking applications), and one for a passive adversary (which can be used for steganography applications). The methods use individual words to embed bits of the secret message. The only information required to read the hidden message is the shared key that was used to embed the message in the first place. Both of the embedding methods spread the modifications evenly throughout the cover text. They also provide the author with the flexibility to exclude parts of the text from being modified by the embedding process. We will discuss these methods in Section 3 and Section 4. See Section 5 for examples of marked text.

Section 6 briefly covers the literature on the evolution of written language use on the Web, as well as the literature on spelling correction, and other relevant work in information hiding.

2. COMPUTATIONALLY ASYMMETRIC EMBEDDING WITH TYPOS

Robust synonym substitution cleverly used sense ambiguity to insert watermarks into text that are resilient against an adversary who uses automated synonym substitution to remove a watermark.[6] This resilience was achieved through the use of *computationally asymmetric transformations* (CAT for short): Transformations (modifications to a cover document to embed a mark) that can be carried out inexpensively, yet reversing them requires much more extensive analyses that necessitates strong artificial intelligence (easy for humans to carry out, but hard to automate).

The task of robust watermarking with typo injection is harder than that using synonym substitution. There are two major difficulties: i) typos are very conspicuous and an obvious target for the adversary (synonyms are stealthy, typos are not) ii) adversaries can use spelling correction tools to undo the effect of embedding. In order to overcome these difficulties we construct CATs with typos. We replace words with typos such that a spelling checker will produce a long list of possible "corrections", hence force the adversary to achieve the capability to understand the underlying text to single out the original word from this list. The resilience of such CATs depend on a multiplicity of alternative typo corrections.

For instance, a spelling checker will easily point the typo in the sentence "Don't forget to bring the ake". However, the correction list will be rather long (ispell version 3.1.18 lists "Abe, ace, age, AK, AK e, AK-e, ale, ape, are, ate, Ave, awe, axe, aye, bake, cake, eke, fake, Ike, jake, lake, make, rake, sake, take, wake" as alternative corrections that are 1 unit edit-distance away from "ake"). Unless the context in which this sentence appears is known, it is hard even for a human to figure out the original word for the typo. Note that some of the alternative words in this list are 2 units edit distance away from the original word "cake". Eventually the adversary is likely to end up choosing a replacement word for the typo, which is further away in terms of edit distance from the original word.

Typos whose correction processes are hard to automate are preferred in robust information embedding. Their correction lists contain several words that have same part of speech and have similar meaning as the original word.

Using the same example as above: an embedding system based on the CAT principle will prefer to use the typo "ake" to the typo "cakw" in the sentence "Don't forget to bring the cake". There are two reasons for this: i) "cakw" has a smaller correction list (ispell version 3.1.18 lists "cake, caw" as alternative corrections for "cakw") ii) while words "ale","sake" (from correction list of "ake") have a similar meaning with "cake" and can be used in the same context, it is unlikely that "caw" (from correction list of "cakw") could be used in this sentence instead of "cake".

Furthermore, CATs can be used to achieve stealthiness of typos, besides achieving resilience. We do this by choosing typos that are themselves legal words from the vocabulary (English vocabulary in this case). This deliberate choice for typos forces the adversary to perform the complex task of detecting such typo words that are used in the wrong context.

The sentence "It's going to be a great party" can be changed to "It's going to be a great patty" by the typo injection mechanism. Since the typo "patty" is also a vocabulary word, a spelling checker will not detect this typo. The adversary does not know which word(s) in this sentence is not from the original sentence. Hence, from the point of view of the adversary, the original sentence is only one of a long list of possible sentences that could have been used to create this watermarked sentence. In order to remove the watermark, the adversary will need to first come up with this long list of sentences (which is expected to include the original sentence before watermarking). Then the adversary will need to make a best guess among these sentences to select the original sentence. This last step of the adversary is similar to the hypothesis ranking problem of automated speech recognition and machine translation.[7] It is known that automated solutions to this problem perform well most of the time in practice. However, in this case, as an important difference from the average case behavior of natural language text or speech, the watermarking process will deliberately choose the typos which will make sure the automated ranking process will perform at its worst. In order to foil the automated adversary, the watermarking system is designed to pick the vocabulary words that create the maximum ambiguity (i.e., longest list of alternative corrections with similar probability).

3. WATERMARKING WITH TYPOS

Without loss of generality, we will describe the method assuming one bit is embedded per word. For now we assume that there are no *untouchable* word occurrences in the text (words that the user forbids to be modified in the encoding process). At the end of this section we generalize the scheme to work for untouchable words as well.

Let V be the vocabulary from which the words in the cover text D are picked. Let K be the shared secret key. Let M be the secret message.

Embedding Algorithm

1. Replace V by another vocabulary V' obtained from V by merging all words, their synonyms and their possible typos (that are not a word from V) into groups, where each group is represented by one (key-selected) word in V'. The representative word that corresponds to w is denoted by $G(w)$ and it is same for all words in a group. For example, {aircraft, airplane, airliner, jetliner, aeroplane, ..., airpane, arplane, airplan, aiplane, ..., aircaft, arcraft, aircrft, aicraft, ..., etc. } are grouped together and only one of them (say, airplane) is chosen as the group's representative. If a word is eligible to appear in multiple groups then ties are broken arbitrarily using the key K to flip a coin.

2. A word token w in D, is denoted by a pair (w, s), if this corresponds to the sth instance of the group represented by $G(w)$ in D.

3. (w, s) is used for information carrying only if the least significant bit of $H_K(G(w)||s)$ is 1, where H_K denotes a keyed cryptographic hash with K as key, and $||$ denotes concatenation. We make sure that any word from the same group will encode the same bit at a given sequence order (by always using the representative of the group to compute the keyed hash). If the adversary uses synonym substitution, the bit value encoded by the word can still be successfully recovered since all synonyms carry the same bit value. If the adversary chooses to inject non-word typos, it is highly likely that the replacement string will be in the same group as the original word. Otherwise there is a 50% chance to flip the encoded bit value.

4. Process each bit m_i in the message M in the following way:

   (a) Scan through the document and find the leftmost information carrying (w, s) in D that has not yet been processed.

   (b) Use the second least significant bit of $H_K(G(w)||s)$ to determine the message bit value carried by (w, s). w already carries m_i with 50% probability; in this case we are done and we move on to embed the next message bit m_{i+1}.

   (c) If w does not carry m_i, then we try injecting different typos into w and collect, in a set of candidates $\mathcal{C}$, the following two types of outcomes for these typo-injections: (i) the resulting typo word $\bar{w}$ is such that the least significant bit of $H_K(G(\bar{w})||\bar{s})$ is 0; (ii) the resulting typo word $\bar{w}$ is such that the least significant bit of $H_K(G(\bar{w})||\bar{s})$ is 1 but its second least significant bit matches m_i. A central technical issue is: Which of the candidates in $\mathcal{C}$ to select? This is tackled separately in the Candidate Selection algorithm given below. For now we note that if a type (i) candidate is selected, $\bar{w}$ is not used to carry any message bits, and in such a case we skip to next information carrying word to encode m_i. If on the other hand, a type (ii) candidate is selected then it corresponds to using $\bar{w}$ to carry the message bit m_i, and in such a case we continue to embed m_{i+1}.

The next algorithm explains how we select from candidate set $\mathcal{C}$ the best alternative.

Candidate Selection Algorithm

1. Partition the typos, that were used to produce the candidate words, into 2 classes: *Conspicuous* typos, and *stealthy* typos. A candidate word $\bar{w}$ that is *not* in the dictionary (such as "imlipcit" instead of "implicit") is considered to be conspicuous. Respectively, a candidate that is in the dictionary (such as "mat" instead of "man") is considered to be stealthy.

   Note that, some of the conspicuous typos of a word are already in the group of the word, and do not change the encoded bit if they are used; the previous steps of the algorithm does not include such words in the list of candidates. The conspicuous typos that are considered at this step come from other groups; they were assigned to another group as a result of a random coin flip when they were eligible for more than one group.

2. If there are candidates in $\mathcal{C}$ whose typo is of the stealthy kind, then we prune $\mathcal{C}$ by removing from it all the candidates whose typo is of the conspicuous kind. Note that the remaining candidates in $\mathcal{C}$ are all stealthy or all conspicuous. In either case, the next step uses the same criteria for selecting the best candidate.

3. For each candidate typo $\bar{w}$ in $\mathcal{C}$, compute the following function $h(\bar{w})$ for it.

   (a) Let $N(\bar{w})$ be the set of *neighbors* of $\bar{w}$: Word a is in $N(\bar{w})$ if the edit distance from a to $\bar{w}$ is no more than a user set threshold, let's say 2. Intuitively, a is in $N(\bar{w})$ if $\bar{w}$ *could* have resulted from a mis-typing of a. In fact the probability of occurrence of such a typo, the conditional probability $\Pr(a|\bar{w})$, is obtained from confusion matrices that quantify the probabilities of various mis-typings.[4] Assume this has been done for all $a \in N(\bar{w})$. Note that the user set threshold can also be given for the $\Pr(a|\bar{w})$, the probability of $\bar{w}$ being the result of a typo in a.

   (b) Having obtained $\Pr(a|\bar{w})$ for all $a \in N(\bar{w})$, we compute $h(\bar{w})$ as follows:

$$h(\bar{w}) = -\sum_{a \in N(\bar{w})} \Pr(a|\bar{w}) \log \Pr(a|\bar{w})$$

   which is the entropy that we should seek to maximize (thereby maximizing the adversary's uncertainty about where $\bar{w}$ could have come from). Note that, here we have assumed that the adversary will only use the knowledge about typo confusion matrices. A more advanced adversary could use n-gram language models, and in that case ambiguity measure of the watermarking algorithm should be changed accordingly.

4. Select from $\mathcal{C}$ the candidate $\bar{w}$ with the largest $h(\bar{w})$.

Typos are injected in a way that maximizes the possible correction alternatives, while staying within a distortion threshold that captures the damage incurred on the cover document after the injection. The distortion threshold is defined by the owner of the document. Viable typos are generated with respect to the confusion matrices used by the spelling checkers, which yields all typos of the cover word that could be typed by a human user. See the next subsection for other objective functions that can be used in watermark embedding.

The decoding is a simple key-based reading of the bit values of the (also key-selected) words in the watermarked text.

We note that extending our system so it can handle 3-letter acronyms such as "lol" would require checking (by pattern matching) whether the current word and its 2 successors (predecessors are not used, since they might have been already used for encoding) can form a 3-letter acronym: If so then replacement by the acronym provides one of the alternative "typos" that we could use.

Extending the above system to include the conversion of non-latin characters into latin characters is straightforward, since such conversions can be treated as typos.

In this method, every typo gives away some information to the adversary about encoding words (i.e., they indicate that one of the dictionary words listed by the spell checker in the correction list is an encoding word). The adversary can use this list to flip some of the message bits by injecting typos into occurrences of all the words in the correction list. We can prevent this damage by inserting unnecessary typos into the watermarked document. This measure also helps to increase the amount of distortion incurred on the document by the watermarking process, hence limiting the error tolerance of the adversary.

Candidate Selection Heuristics

The heuristic introduced in step 3.*b* of the Candidate Selection Algorithm maximizes the possible correction alternatives for encoding words. This increases the uncertainty about the original versions of encoding words, and make it harder for the adversary to revert the watermarked text back into original form.

We can alternatively use another heuristic to maximize the probability that the encoded bit will stay the same even if the adversary chooses to randomly replace the encoding word:

For all the candidate words that are encoding the desired bit m_i, pick the information carrying neighbors of $\bar{w}$ that encode m_i and put them into set $N'(\bar{w})$, then compute $\Pr(m_i|\bar{w})$ as follows:

$$\Pr(m_i|\bar{w}) = \sum_{a \in N'(\bar{w})} \Pr(a|\bar{w})$$

Then pick the candidate word that maximizes $\Pr(m_i|\bar{w})$. Here we seek to maximize the probability that adversary will pick a word that is still encoding m_i even though s/he replaces $\bar{w}$ with a word from its neighbors $N(\bar{w})$.

Extension to Untouchable Words

One of the disadvantages of information hiding in text is the low bandwidth of the medium. Since the document units (e.g., words, phrases, etc.) that carry the embedded bits is scarce, every available unit is used for message carrying. Cursory text tolerates such saturation of the text with modifications for most of the time. However, it is not uncommon to have parts of the text which are too sensitive and modifications to the original are not tolerable (such as a date, a salary figure, a military rank). It is important that information hiding systems have a way of accommodating "untouchable document areas" by avoiding modifications to such portions.

It is possible to avoid untouchable areas if "untouchable words" are known beforehand, by skipping the individual words or the phrases and sentences that they occur. However this is not a practical approach for a general purpose information hiding system.

Untouchable words are like the "defective memory cells" in Wet-Paper Codes (WPC), and hence efficient wet-paper codes can be used to handle them[8] for text steganography, see Section 4 for more details.

In our algorithm, "untouchable words" are problematic only if the word is used to carry the watermark (i.e., least significant bit of $H_K(G(w)||s)$ is 1). An alternative encoding could easily solve this problem when untouchable words are rare and isolated (as opposed to occurring in chunks of text): Encode a message bit m_i (i.e., the bit value carried by jth information carrying word w_j in D) as the XOR of the *third* least significant bit of $H_K(G(w_{j-1})||s_{j-1})$ with the second least significant bit of $H_K(G(w_j)||s_j)$.

This may entail having to backtrack and modify w_{j-1} for the sake of encoding bit m_i in w_j while still satisfying the old requirement imposed on w_{j-1} by its encoding of m_{i-1} or of the (non)information carrying property of w_{j-1}. The drawback of this approach is that it will cause a substantial decrease in the number of candidates in $\mathcal{C}$ for w_{j-1} (approximately by a factor of two), and hence a lower conditional entropy for the chosen candidate from that $\mathcal{C}$. Of course no such burden is imposed on w_{j-1} if w_j is not untouchable. Note that the bandwidth is also largely unchanged – it does *not* go down by a factor of 2 because, when we get to m_j, we do not need to backtrack to m_{j-1} because the 3rd bit of $H_K(G(w_{j-1}), s_{j-1})$ is already fixed by then and its presence in the XOR makes no difference to the success (or lack thereof) of encoding m_i in w_j. Also note that the bandwidth would have dropped by a factor of 2 if we had used the second bit of $H_K(G(w_{j-1}), s_{j-1})$ in the XOR.

See Section 5 for an example.

4. STEGANOGRAPHY WITH TYPOS

In steganography, stealthiness of the marked document is a more important concern than the robustness of embedding against modifications by an adversary or the value of the cover document. Here the adversary is passive and only interested in detecting the covert communication.

We use the same notation that the watermarking algorithm uses in Section 3: V is the vocabulary from which the words in the cover text D are picked; K is the shared secret key; M is the secret message.

Let T be the user defined total distortion threshold. We define distortion, $d()$ to be the probability of a typo word, $\bar{w}$, occurring in a text as a result of a typing error made by a human typist that intended to write the original word, w. Hence $d(\bar{w}, w) = \Pr(w)\Pr(\bar{w}|w)$. We use the spelling error probability metric defined by Kernighan et al. in.[4] See Section 6 for more details on this spelling error correction method. The steganography process ensures that $\sum_{w \in D} d(\bar{w}, w)$ stays below T. Note that this distortion measure assumes an adversary would

know the original word w, and use this information to detect a steganography. Hence the adversary model of the steganography system we describe in this section is more sophisticated than an automated system.

We propose a steganography system that has the flexibility to let the user forbid some of the words from being modified. Such words are considered "untouchable" and the marking process finds a way to encode the secret message without touching these words in the cover text. This way, users may i) conserve the value of the cover document from being destroyed by the embedding process, ii) achieve stealthiness to the human eye, which would not tolerate extensive typing errors at critical words. Usually, preserving the value of the cover document is not a primary concern in steganography. However, there might be automated and manual filtering systems that have been installed to remove the documents that do not serve the purpose of the communication channel (e.g., spam filters, or a collaborative filtering system in a forum in which other users vote to remove posts with no relevant information).

Our steganography algorithm uses efficient Wet-Paper Codes (WPC) technique presented in[8] to achieve minimum distortion while allowing the user to mark certain sections of the cover text as untouchable. These untouchable sections will act as "defective memory cells" in WPC.

Let L denote the sections that are untouchable.

Embedding Algorithm

1. Replace V by another vocabulary V' obtained from V by merging all words and their possible typos into groups, where each group has a (key-selected) *representative* word for that group. For example, {word, owrd, wrod, wodr ...etc. } would be grouped together and only one of them is chosen as the group's representative. The representative word that corresponds to w is denoted by $G(w)$ and it is same for all words in a group. If a word is eligible to appear in multiple groups then ties are broken arbitrarily using the key K to flip a coin.

2. A word token w in D, is denoted by a pair (w, s), if this corresponds to the sth instance of the group represented by $G(w)$ in D.

3. (w, s) is used for information carrying only if the least significant bit of $H_K(G(w)||s)$ is 1, where H_K denotes a keyed cryptographic hash with K as key, and $||$ denotes concatenation.

4. Read the bit string that D carries (before the message embedding) into the string B as follows:

   (a) For each information carrying word w in D, assign the second least significant bit of $H_K(w||s)$ to b_i in B. (Note that the word w is directly included in the hash in this case.)

5. Use the efficient wet-paper code technique[8] to generate a set of bit strings, $\mathcal{B}$, that can encode M without using the bits that fall into the sections marked by L and whose hamming distance to B is below a threshold (this threshold is set a priori).

6. For each $B' \in \mathcal{B}$ use the embedding transformations defined in the watermarking algorithm in Section 3 to embed B' into D and get the marked text D'. The only difference here is the candidate selection criteria: while picking a typo from the possible candidates in the set $\mathcal{C}$, we pick the candidate, $\bar{w}$, that is the most probable typo (i.e. $\bar{w} = \arg\max_{w} \Pr(\bar{w}|w)$) in the list of typos that can encode the desired bit, m_i.

7. Pick the B' that inflicts minimum total distortion on D, and output the corresponding stego text, D' that carries B'.

The decoding process is similar to watermark reading, where the bits encoded by the key-selected words in the stego text is read using the keyed hash.

See Section 5 for an example.

5. EXPERIMENTS: MAKING OF MARKERR

The current implementation of MarkErr has two main characterizing inputs: i) the vocabulary of words ii) distortion measure. The vocabulary determines which words are eligible to be considered for message carrying. The vocabulary also determines the words that can be used as replacement words by the watermarking algorithm. The user can limit the choice of words that can be injected to the cover document (i.e., by removing such words from the vocabulary if they are inappropriate). We used the master English word list of Aspell Version 0.50.5 as the vocabulary in our implementation.

We use two different distortion measures to quantify the cost of a unit transformation of the cover text for watermarking and steganography. The distortion measure of steganography is based on a model of typos that have been extracted from AP newswire text in 1988.[4] It is $-log(P(t|w))$, logarithm of the probability of a particular error given the correct word. A stego embedding with a lower cost should be similar to a human-made typo, hence be less detectable by an adversary.

The watermarking distortion measure should quantify the distortion of the value of the text for human readers. Even though we cannot perfectly capture this as a number, we believe that the similarity of the alternative corrections of a typo may capture the confusion of the readers when faced with a typo and hence approximate the degradation in the value of the cover text. The entropy, $h(\bar{w})$ in step 3.b of the *Candidate Selection Algorithm* in Section 3 quantifies this confusion of the user. In this implementation, we did not use context information when computing the entropy of the correction word (i.e., an injection of **hat** to replace **that** is less confusing than when **hat** is used to replace **cat**). A more sophisticated implementation should use side information (e.g., language models, word clusters, part of speech, etc.) to determine words which are commonly used in similar contexts and can be accidentally typed in place of each other.

Following example shows a watermarked version of the first three sentences of the abstract of this paper. We embedded 16 bits into this text. Note that the embedding transformations were performed by using "stealthy" typos where the mark carrying word is a vocabulary word. (Changed words are shown in bold font.):

> A substantial portion of the text available online is of a kind that tends to contain **mane** typos and ungrammatical abbreviations, e.g., emails, blogs, forums. It is therefore not surprising **chat**, in **suck** tests, one can **tarry** out information-hiding by the judicious injection of **tyros**. The resilience is achieved through the use of *computationally asymmetric transformations* (CAT for short): Transformations that can be **married** out inexpensively, yet reversing them requires much **mere** extensive semantic analyses (easy for humans to carry out, but **hark** to automate).

In our implementation for steganography we used the conspicuous typos as embedding transformations. These typos are generated by applying letter changes to the original words in the cover text which usually yield non-vocabulary tokens in the marked text. Even though these transformations can easily be detected by the adversary, they still have the CAT property as the adversary has to find out the original correct wording of the typographical error among many viable alternatives in order to revert the transformation. For instance a deletion transformation of "change" into "chage" will force the adversary to choose one of "achage, cage, chafe, Chaga, Chane, change, chape, chare, charge, chase, cha ge, cha-ge, phage" to revert the typo. An example of this embedding is shown below:

> A substantial **potion** of the text available **onlne** is of a kind that tends to **conain** many typos and ungrammatical abbreviations, e.g., **emals**, **blgs**, forums. It is therefore not **surprisng** that, in **sich** texts, one can **carrsy** out **infomation-hiding** by the judicious injection of **tpyos**. The resilience is achieved through the use of *computationally asymmetric transformations* (CAT for **hsort**): Transformations that can be carried out inexpensively, yet reversing them **requirs** much more extensive semantic **analsyes** (**esasy** for **humas** to **carrsy** out, but **harsd** to **uatomate**).

The original text is the same as the watermarking example (first three sentences of the abstract) and again we embedded 16 bits into this text.

6. RELATED WORK

In her 1995 book,[9] Turkle published the results of her analysis on the behavior of users in a multi-user game (e.g. Multi-User Dungeon (MUD)) that allows players to chat. She states that onomatopoeic expletives and a relaxed attitude toward sentence fragments and typographic errors suggest that the new writing is somewhere between traditional written and oral communication. This type of language used online is commonly referred as "NetSpeak".[10] While Turkle's focus is mainly on the psychological effects of the Internet environment, Crystal's work focuses on analyzing the evolution of the language used in chatgroups, emails, and text messages send over mobile phones.[11,12] Crystal mentions that due to the 160 character limitation in text messages, users tend to shorten many words, and these acronyms or abbreviations stick in a community. Crystal also discusses the ambiguity in NetSpeak in his article.[12] He mentions that there is a two way ambiguity in this language, one way is while interpreting what an acronym stands for, for example "N" can mean "no" and "and", "Y" can mean "why" or "yes". It is up to the receiver to decode a sender's message when it involves an ambiguous acronym, "GBH" can mean "great big hug" or "grievous bodily harm". The other way of ambiguity occurs when shortening a term, for example, "good to see you" can be "GTCY", "GTSY", "G2CY" or "G2SY", and "thanks" can be "THNX", "THX", "TX" or "TNX". Even though usage of acronyms are two-way ambiguous, embedding information through them forms CAT type of a transformation, where computational complexity of forming an acronym out of a word or a phrase is much more lower than disambiguating the meaning of an acronym.

The studies in spelling error detection and correction research have focused on three problems[7,13]:

- **Non-word Error Detection** This problem involves finding out whether a word is not in a given dictionary. Several efficient pattern matching and *n-gram* analysis techniques have been proposed for solving this problem, which requires correctly parsing a given word into its stem and its suffix; a fast search capability and a well-designed dictionary. The Unix ®*spell* program is one of the commonly used non-word error detection tools.

- **Isolated-Word Error Correction** Analysis of word typing errors occurring in several applications such as newswire text, search engine queries or optical character recognition data has shown that error rate and error type (e.g. single or multiple character errors) varies from application to application. There have been several techniques designed for detecting and proposing corrections for a misspelled word, such as minimum edit distance technique, rule-based techniques, n-gram based techniques, or probabilistic techniques–such as the one we have used in our experiments.[4] All of these techniques have proven to be successful in a given domain, but an isolated word error correction technique that works efficiently for any given domain has not yet been introduced.

- **Context-Dependent Word Correction** This problem involves dealing with errors where an actual word is substituted for another actual word. This can happen in many forms: due to typos (e.g typing "form" instead of "farm", "lave" instead of "leave"); due to cognitive or phonetic mistakes (e.g. "there" instead of "their", or "ingenuous" instead of "ingenious" or typing "f" instead of "ph"); due to use of wrong function word (e.g. "of" instead of "on"); due to improper spacing (e.g. "my self" instead of "myself"); due to insertion or deletion of whole words (e.g. "I cut myself when while cooking."); due to grammar errors (e.g. "he come" instead of "he comes") etc. Devising a solution for correcting this type of errors requires strong natural language processing capabilities including the challenging topics of robust natural language parsing, semantic understanding, pragmatic modeling and discourse structure modeling. Only a few spell correction tools attempted to perform context-dependent word correction, and so far none of them have been successful in solving this problem beyond a domain dependent setting that allows only a very restricted type of errors (e.g. at most one misspelled word per sentence, each misspelling is the result of a single point change, and the relative number of errors in the text is known).[14]

As also mentioned in Section 5, while implementing MarkErr, we used the probabilistic spelling correction technique introduced by Kernighan et al.[4] This technique uses a Bayesian argument that one can often recover the intended correction, c, from a typo, t, by finding the correction that maximizes $\Pr(c)\Pr(t|c)$. $\Pr(c)$ is the word probability learned from a corpus by using the frequency count of a word, and $\Pr(t|c)$ is a model of the

noisy channel that accounts for spelling transformations on letter sequences (i.e. insertion, deletion, substitution and reversal). There is one confusion matrix for each spelling transformation. This confusion matrix shows the probabilities of the transformation occurring between the two letters such as $count(sub(t_p, c_p))/count(chars(c_p))$ shows the probability of a character c_p being substituted by t_p. The matrix for $\Pr(t|c)$ is computed using the four confusion matrices computed for each spelling transformation. Kernighan et al. used Associated Press Newswire corpus for training these probability models. Given the typo word "acress", this spelling correction method (when trained on AP newswire corpus) produces the following list where the correction words are sorted according to their scores: {acres (0.45), actress (0.37), across (0.18), access (0.00), caress (0.00), cress (0.00)}. Confusion matrices for all four spelling transformations are provided in.[4]

Besides the above mentioned challenges spelling correction for cursory text – similar to spelling correction for search engine queries[14] – has unique challenges such as maintaining a dynamic dictionary which should be updated to include terms emerging in daily life: acronyms (e.g. "asap", "lol"), emoticons (e.g.":-D"), new terms (e.g "blogging", "googling", "phishing", "pwned"), uncommon person names (e.g. "Suri", "Shiloh"), newly generated words for marketing purposes (e.g. a recent movie directed by Gabriele Muccino is titled "The Pursuit of Happyness", one of the popular songs performed by Avril Lavigne is titled "Sk8er Boi"). Such requirements make devising a highly accurate spelling correction tool for cursory text very hard.

Most of the studies in information hiding into natural language text is based on re-writing the cover document using linguistic transformations such as synonym substitution,[15,16] or paraphrasing.[1,2] T-Lex is one of the first implemented systems that embed hidden information by synonym substitution on a cover document.[15,17] T-Lex first generates a database of synonyms by picking the words that appear only in the same set of synonym sets from WordNet. The intersections between distinct synonym sets are eliminated to avoid usage of ambiguous words for encoding. This filtering causes the use of uncommon words (e.g. replacing "nothing" with "nada") due to the fact that common words tend to span through several unrelated synonym sets and this property can easily be exploited by steganalysis techniques that use language modeling such as the one introduced in.[18]

In,[17] Bergmair provides a survey of linguistic steganography. He also discusses the need for an accurate word sense disambiguator for a fully automated synonym substitution based steganography, where sense disambiguation is required both at decoding and encoding time. The lack of accurate disambiguation forces the synonym substitution based information hiding systems to restrict their dictionaries to a subset of words with certain features. Besides decreasing the communication bandwidth, such restrictions cause the systems to favor use of rare words for encoding information.[18]

In another work, Bergmair et al. proposes a Human Interactive Proof system which exploits the fact that even though machines can not disambiguate senses of words, humans can do disambiguation highly accurately.[19]

Topkara et al. have recently designed a lexical watermarking technique[6] and build it into a system, Equimark, that achieves good embedding and resilience properties through synonym substitutions. When there are many alternatives to carry out a substitution on a word (that was selected as a message carrier), Equimark prioritizes these alternatives according to their ambiguity, and uses them in that order. Besides having this one-wayness feature, Equimark allows the owner of the document to set a distortion threshold. Embedding process stays within this threshold, while maximizing the expected distortion that has to be applied by an adversary that is trying to remove the embedding. Equimark uses a weighted undirected graph of (word, sense) pairs, where an edge between two nodes represents that they are synonyms and the weights on the edges are assigned according to the similarity between two adjoining words .

7. CONCLUSIONS

We have presented a robust information hiding system that is based on the clever use of idiosyncrasies (such as typing errors, use of abbreviations, and acronyms) that are common to cursory text (e.g, e-mails, blogs, forums).

We use computationally asymmetric transformations (CAT), that are computationally inexpensive to perform but hard to revert back (without disproportionately larger computational resources, or human intervention), such as replacing a word with a typo that has a long list of equally possible corrections.

We have designed and implemented two different systems, one for watermarking (robust against an active adversary) and one for steganography (stealthy against a passive adversary).

The language of cursory text is evolving, and getting richer by new acronyms (e.g, "lol") or new words (e.g., "phishing") added daily to the language by repeated usage in many online communities. There is much room for improvement in information hiding in cursory text. Typing error is only one type of idiosyncrasy that opens room for information hiding in cursory text, and new opportunities develop as the language develops.

REFERENCES

1. M. Atallah, V. Raskin, C. F. Hempelmann, M. Karahan, R. Sion, U. Topkara, and K. E. Triezenberg, "Natural language watermarking and tamperproofing," *Proceedings of the Fifth Information Hiding Workshop*, vol. LNCS 2578, 7-9 October 2002, Noordwijkerhout, The Netherlands.
2. M. Topkara, U. Topkara, and M. J. Atallah, "Words are not enough: Sentence level natural language watermarking," *Proceedings of ACM Workshop on Content Protection and Security (in conjuction with ACM Multimedia)*, October 27, 2006, Santa Barbara, CA.
3. G. Tur, "Turkish text de-asciifier," *http://www.hlst.sabanciuniv.edu/TL/deascii.html*, 1998.
4. M. D. Kernighan, K. W. Church, and W. A. Gale, "A spelling correction program based on a noisy channel model," *Proceedings of the 13th conference on Computational linguistics*, 1990, Morristown, NJ, USA, Association for Computational Linguistics, pp. 205–210.
5. L. Philips, "Hanging on the metaphone," *Computer Language Magazine*, vol. 7, no. 12, pp. 39–44, 1990.
6. U. Topkara, M. Topkara, and M. J. Atallah, "The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions," *Proceedings of ACM Multimedia and Security Workshop*, September 26-27, 2006, Geneva, Switzerland.
7. D. Jurafsky and J. Martin, *Speech and Language Processing*. Upper Saddle River, New Jersey: Prentice-Hall, Inc, 2000.
8. J. Fridrich, M. Goljan, and D. Soukal, "Wet paper codes with improved embedding efficiency," *IEEE Transactions on Information Forensics and Security*, vol. 1, pp. 102–110, March 2006.
9. S. Turkle, *Life on the Screen: Identity in the Age of the Internet*. New York: Simon and Schuster, 1995.
10. L. Truss, *Eats, Shoots & Leaves*. New York: Gotham Books, 2004.
11. D. Crystal, *Language and the Internet*. Cambridge CUP, 2001.
12. D. Crystal, "Txt, ny1?," *In Susan Tresman and Ann Cooke (eds), The Dyslexia Handbook*, 2006, British Dyslexia Association, pp. 179–183.
13. K. Kukich, "Technique for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, 1992.
14. S. Cucerzan and E. Brill, "Spelling correction as an iterative process that exploits the collective knowledge of web users," *Proceedings of EMNLP 2004*, 2004, pp. 293–300.
15. K. Winstein, "Lexical steganography through adaptive modulation of the word choice hash," *http://www.imsa.edu/ keithw/tlex/*, 1998.
16. M. Atallah, C. McDonough, S. Nirenburg, and V. Raskin, "Natural Language Processing for Information Assurance and Security: An Overview and Implementations," *Proceedings 9th ACM/SIGSAC New Security Paradigms Workshop*, September, 2000, Cork, Ireland, pp. 51–65.
17. R. Bergmair, "Towards linguistic steganography: A systematic investigation of approaches, systems, and issues.," tech. rep., University of Derby, November, 2004.
18. C. M. Taskiran, U. Topkara, M. Topkara, and E. Delp, "Attacks on lexical natural language steganography systems," *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, 2006.
19. R. Bergmair and S. Katzenbeisser, "Towards human interactive proofs in the text-domain," *Proceedings of the 7th Information Security Conference*, vol. 3225, September, 2004, Springer Verlag, pp. 257–267.